Project Design Phase-II
Technology Stack (Architecture & Stack)

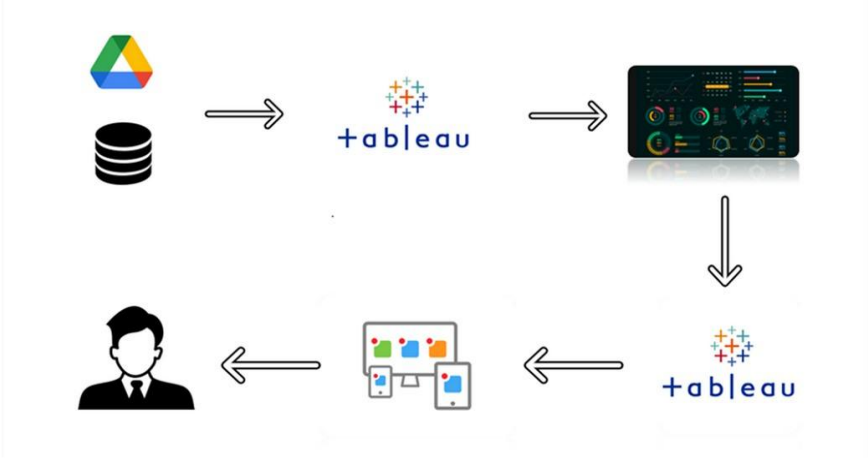
Date	03 November 2023
Team ID	NM2023TMID06902
Project Name	iRevolution: A Data-driven Exploration of Apple's iPhone Impact in India
Maximum Marks	4 Marks

Technical Architecture:

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2

Example: Order processing during pandemics for offline mode

Reference: https://developer.ibm.com/patterns/ai-powered-backend-system-for-order-processing-during-pandemics/



Guidelines:

- Include all the processes (As an application logic / Technology Block)
- Provide infrastructural demarcation (Local / Cloud)
- Indicate external interfaces (third party API's etc.)
- Indicate Data Storage components / services
- Indicate interface to machine learning models (if applicable)

Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	How user interacts with application e.g. Web UI, Mobile App, Chatbot etc.	HTML, CSS, JavaScript / Angular Js / React Js etc.
2.	Application Logic-1	Logic for a process in the application	Java / Python
3.	Application Logic-2	Logic for a process in the application	IBM Watson STT service
4.	Application Logic-3	Logic for a process in the application	IBM Watson Assistant
5.	Database	Data Type, Configurations etc.	MySQL, NoSQL, etc.
6.	Cloud Database	Database Service on Cloud	IBM DB2, IBM Cloudant etc.
7.	File Storage	File storage requirements	IBM Block Storage or Other Storage Service or Local Filesystem
8.	External API-1	Purpose of External API used in the application	IBM Weather API, etc.
9.	External API-2	Purpose of External API used in the application	Aadhar API, etc.
10.	Machine Learning Model	Purpose of Machine Learning Model	Object Recognition Model, etc.
11.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration: Configuration for deploying the application on local servers. Cloud Server Configuration : Configuration for deploying the application on cloud platforms like Local, Cloud Foundry, Kubernetes, etc. Infrastructural Demarcation: Local, Cloud	Local, Cloud Foundry, Kubernetes, etc.

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	● Web Framework ● Front-end Framework ● Backend Framework	● Express.js (Node.js) ● React ● Backend Framework: Django

		• Database Framework	(Python) • Hibernate (Java)
2.	Security Implementations	• AES-256 for data encryption at rest and in transit • Implement IAM roles and policies for AWS services • Implement OWASP (Open Web Application Security Project) best practices to mitigate common security vulnerabilities • Use firewall rules and security groups to control incoming and outgoing traffic • Use SHA-256 for hashing sensitive information	• Encryption • Identity and Access Management (IAM) Controls • Web Application Security • Firewall • Secure Hash Algorithm (SHA-256)
3.	Scalable Architecture	• Our application is designed with a 3-tier architecture, separating the presentation, logic, and data layers for better scalability and maintainability. • We employ a microservices architecture, allowing us to scale individual components independently, ensuring high availability and resource optimization. • Implement load balancing with technologies like AWS Elastic Load Balancing or Nginx to distribute traffic evenly across multiple instances. • Use containerization tools like Docker and orchestration platforms like Kubernetes to manage and scale microservices.	• 3-Tier Architecture • Microservices • Load Balancing • Containerization

S.No	Characteristics	Description	Technology
4.	Availability	• Implement load balancers to evenly distribute user requests across multiple servers, ensuring redundancy and fault tolerance. • Employ a multi-server setup to distribute application components, reducing the risk of downtime due to server failures. • Use auto-scaling mechanisms to dynamically adjust server resources based on varying traffic demand, ensuring consistent performance during peak usage. • Implement failover and backup strategies to minimize downtime and data loss in case of server or component failures. • Utilize database replication and clustering solutions to ensure continuous data availability.	• Load Balancers • Distributed Servers • Auto-Scaling • Failover Mechanisms • High Availability Databases
5.	Performance	• Implement caching mechanisms using technologies like Redis or Memcached to store frequently accessed educational content, reducing server load and improving response times. • Content Delivery Network (CDN): Utilize CDNs like Cloudflare or Akamai to cache and serve educational resources from edge servers closer to students, reducing latency and improving access speed. • Perform load testing with tools like Apache JMeter or Gatling to determine the application's capacity to handle	• Caching • Content Delivery Network (CDN) • Load Testing • Adaptive Learning Algorithms • Interactive and Responsive User Interface

		concurrent students and identify performance bottlenecks. ● Adaptive Learning Algorithms: Employ machine learning models to personalize and adapt learning experiences for individual students, improving engagement and outcomes. ● Interactive and Responsive User Interface: Develop a responsive user interface with technologies like Angular, React, or Vue.js to ensure a seamless and enjoyable user experience for students.	

References:

https://c4model.com/

https://developer.ibm.com/patterns/online-order-processing-system-during-pandemic/

https://www.ibm.com/cloud/architecture

https://aws.amazon.com/architecture

https://medium.com/the-internal-startup/how-to-draw-useful-technical-architecture-diagrams-2d20c9fda90d